John Seely Brown wants to support
author's Deep Structure!
His Authorship proposal

HT 1987)

SAUSALITO, CA 94965

~887
[MCC]

CIS working paper

Notes concerning desired functionality, issues and philosophy for an *AuthoringLand*

John Seely Brown

February 16, 1982
revised--june
revised (slightly)--july

Theme: From Chaos to Order--A way to view writing

(and some writing tools to help the author discover structures)

1) Premise: Nearly all past editing systems, even those calling themselves "structural editors" address only the issues of helping the author craft the *final* prose and perhaps the layout of the final document. Claim: this aspect of writing is like the tip of an iceberg, reflecting, at most, 10% of one's actual effort in writing a paper. report or book. If we are to make progress in creating significantly more powerful authoring aids we must break away from our institutions, belief structures and biases that are grounded in tools for crafting the *surface* structure of a document--i.e. its form or its appearance.

From our perspective, AuthoringLand is, perhaps, more like an IdeaLand. It is meant to help us move from **Chaos to Order** in our thinking and writing endeavors. It should help us move from our pre-articulate intuitions to articulate ideas and from there to structuring the ideas for presentation, a structuring highly dependent on the purpose, form (e.g. pape, slides etc) and intended readership.

>analogy: We want tools that help authors discover, articulate, organize and effectively communicate ideas. As such, our concerns here relate more to the kinds of tools being developed for powerful AI programming enviroments--namely environments that help the programmer actually unfold or discover what the real problem is (or what the algorithm must be to solve it) in contradistinction to those tools developed for structured programming where the task is to "solve" a completely defined problem or to implement an already

undersood algorithm. Said differently, in progrmaming, the tools that help us the most in transforming an ill-defined problem into a defined problem are apt to be quite different from those that help us craft and prove correct an implementation of an algorithm for solving a well-defined, well specd problem. (Thus enters the misguided controversy between Interlisp vs Pascal, for example.)

>tools to help circument writer's block or moving from chaos to order:

Conjecture--a blank piece of paper instills blockage because we view the writing process not as a stream of consciousness but rather as being a highly ordered process. (For example, consider the way either Collins or I write proposals.) When we first set out to write we can't possibly have all our ideas structured or even ordered. What we need are tools and a medium to help us discover this structure and help us move from our pre-articulate intuitions to articulatable ideas. This is where the payoff can be and it is surely being ignored, if not defeated, through the design of "structured editors" or such tools that require you first to specify an "outline" and then progressively refine/expand that structure etc. At some point in the writing prcess that might be useful but for many of us it would be awful to have to begin writing a paper by first having to specify an outline.

Thus---

2)Tools will have to differ depending on:

i) the current degree of chaos (or stage) in our thinking/writing process. For example, when we first start out to define, refine and perhaps discover our ideas, we will need tools for massaging, organizing, browsing something more akin to a stream of consciousness than a carefully thoughtout structure. After the ideas, caveats, achille heels, counter-ideas etc are articulated then we need tools to help create some structure over them for presentation. Such tools should reflect, or atleast facilitate, rheotorical (compositional) constructs/theories/issues and perhaps even request information from us concerning the intended style, readership etc. Following that we will need tools to help us craft the actual "surface" document. Here the tools will resemble those associated with document compilers, spelling correctors, proofcheckers, index and cross-reference generators, figure chart graph footnote generators etc.

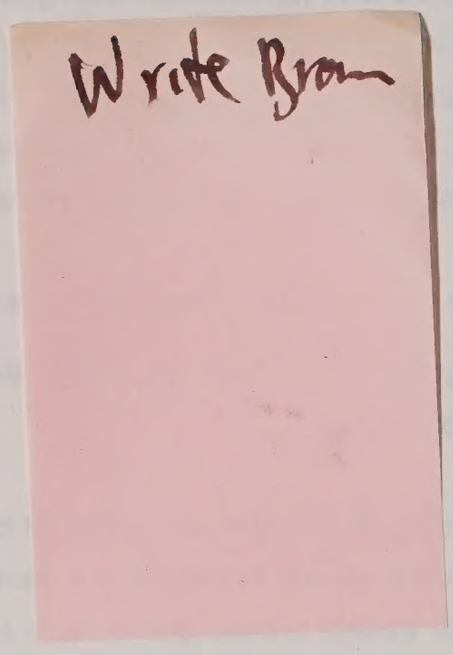
ii) the intended or actual size of the document (i.e. short note, journal article or book--the latter requiring, perhaps, many more powerful organizational and reminding aids.)

iii) the number of different kinds of uses of the doc-*space*. For example, from the same document/conceptual space I might want to construct slides for a talk, simple summaries for progress reports, informal notes for a colleague and a scientific (ho ho) paper. As I am working on developing any one of these, or just developing my stream of consciousness, I may want to jot down something relevant for another use. Also, and more to the point, I might want to try out radically different organizations of the ideas/points for, say, generating transparancies than for a paper. (This is especially true in that I often use a talk as a chance to "float" a new balloon in terms of trying out a radically new way to present, view, organize a piece of ongoing research.)

iv) how many authors (or workers) there are for the document. In partiuclar, one of the forces stimulating my interest was the **total lack** of any tools to help co-authoring a paper. (Incredible for 1982)

>For example, at a minimum we should be able to have all the changes and/or additions that one author makes to the current working doc be highlightable for the other authors to be able to quickly browse and scream [many technical problems here concerning lockouts, mergers etc]-although I suppose such that a feature couid be useful for even one author so that he could see what he had changed during the previous session. More on that later since that concerns the notion of contexts and layers in a doc etc.

Presupposition--we will never know all the capabilities that we will need or even how to deploy those that we have already created thus what we really need, at a minimum, is a programmable user-profile and perhaps, also a truly programmable AuthoringLand, itself. Note that Emacs type editors don't cut it although perhaps they are better than anything else around. Nevertheless Emacs and its offsprings are still chiefly *cognitively impenetrable*. (This raises a serious issue: perhaps we don't have enough power yet to write a system that is both systematically structured (and hence understandable and modifable) and efficient enough to feel "right"--whatever that means but note that since this system will involve managing multiple visual objects, speed will be important. There are so many features one would like even this base editor to have that the need to have it programmable is all too clear. But nevertheless we will need to examine the class of capabilities that we, the users, might want to be able to program it to have.


Write Bran

3) Implementations: (concerning the text editor)

i) For the low level editor and perhaps a rough doc layout system, we would like to be able to think about its behavior as being defined by a set of *constraints* that govern, for example, how and when to scroll up when one is madly typing away overflowing the page, how to handle footnotes. how to merge figures, how to prevent widows and rivers etc. In my initial flametude I thought we might be able to build the whole system around a constraint engine coupled to a truth maintainence system (TMS) that would enable the doc that already satisfies all the constraints to be locally changed with the TMS mechanism taking care of retracting the minimal set of propogations in order to efficiently re-satisfy all the constraints. If this could be done, the the user might be able to declaratively specify the aesthetic principles/constraints that he wants or atleast could modify those already handed him on the silver plate. After all, I doubt that even Tex has that complex a "theory" of how to justify text and how to layout a doc. Note that if this theory could be made absolutely explicit as a set of assertions with declarative import, then this could act also as an interesting educational tool to help get kids/users more aware of the subtle and large amount of tacit knowledge that we all take for granted in what makes a document look right.

ii) Probably would like to have a mixed or intermdieate representation of the doc space where one could specify commands like {make italics the following...} by just doing it a la Bravo and in so doing appropiate functions or whatever are wrapped around the expression in the underlying representation and at the same time the stuff on the screen is put into italics. This, of course, transforms the naive realism philosphy from "what you have is what you see" to "what you have is more than what you see unless you really want to "see"".

iii) We should recognize that each of us, as we attempt to type, have a set of rather idiosyncratic typing/spelling errors. Maybe we could let the user build up a set of macros for efficiently correcting his unique of errors. (For example, one might want a feature that enables two letter sequences to be flipped as one action. Something akin to this can happen with the use of the buffers in Bravo and the escape key but it is limited and implemented as a sequence of actions which takes time especially given that Bravo will try to rejustify the "paragraph" in the middle of the action.)

> Note that this brings up another major issue. It will be all too easy to have either the system designer of such an editor/doc-compiler/authoringland or the user, himself, create gigantic number of random features that exceed anyones ability to learn,

remember and use them all. We need to think of some kind of facility to help the user browse and utilize the current set of features. This suggests that we might want something akin to our old notions about using Klone to make explicit the current features and requirements of those features etc. Such a structure could be browsed, summarised, expanded on demand and serve as a guide/backbone for adding more capabilities.

>Likewise, if we went the KL-ONE route, it would also open up the possibility of using the network to structure the objects that might be referred to in constructing a rule-based system of "intelligent" agents that would provide another kind of flexiblility to an AuthoringLand. This links us back into the intriguing notion of giving the system **standing orders** concerning, for example, how to handle the *remind* list/agenda or the message component etc--more on this later.

[Another kind of command I would like to be able to define is a command the would capitalize what ever is selected. This suggest that we will also need soft or loadable keys/buttons a la our old CompassLand where we have a menu that not only shows the possible commands but also graphically shows the current "binding" of commands to the three buttons on the mouse.]

4) More random comments on the low-level editor:

>The FIND type commands:

>want it to be able to cross "<cr>" boundaries especially if the user operates in the mode of typing <cr>s etc.

>want a rich pattern specification scheme that can specify what I *don't* want in the surrounding context as well as what I do want. Probably also a programming-by-example scheme with the user giving explict advice on how to generalize the desired and undesired context a la Mike William's reformulation stuff?.

>when a token matching the pattern is found, we might want it to be able to highlight the entire sentence that contains it. This feature helps on doing multiple finds or substitutes where the system finds the next occurence of the specified token and asks the user if he wants the transformation to be applied. In such cases the user often needs to be able to quickly identify the entire sentential context.

>>there are various heuristics for finding such boundaries without needing a parser...however with LFG around the corner, why not be brave and go for broke.

>>>might want a parser not only for all the great proofreading ideas that have been kicked around (note that there are still many kinds of proofreading checks/heuristics that can operate without a fully general parser and that may still catch lots of *editing* mistakes since such mistakes often leave the sentence badly mutilated) but also for **cogntively justified** text. That is, a text justifer that pushes text to the next line based on phrase boundary considerations. {need to find out more about Bever's experiments that show how much faster one can read text that has been justified accordingly.}

[Note the above two typing errors oiut' and emtsn'. Each of these has reoccured many many times in writing this brief doc. Each has an obvious basis to it and might lend itself to a special command to rectify it or suggest some neuro surgery to help me ouit. On the other hand, one might just leave all such errors in the doc and wait for a spelling corrector to fix it. Or, perhaps, we would like to have two levels of spelling correction: one more heuristic that wouldn't need to be hooked into a dictionary server (so I could use it at home and that would be fast) and one that counted on having access to a complete dictionary.

>**Redo** and **History** lists: Will want to have al the kinds of features that Inierlisp-D currently supplies concerning Redo, Undo etc.

More on Authoring as opposed to boring editing:

5) Co-processing aids: When we are in the process of writing. ideas often pop-up pertaining to things that need to be done relating i) to the current idea being worked on, ii) to other ideas in the paper, iii) to supporting docs that one desparately needs, iv) to issues unconnected to the current paper like--christ, I have never responded to HJ about his draft proposal on authoring, etc.

i) We need an **active scratch pad** that enables us to jot stuff down that we don't want to spend the cognitive resources on, at that moment, even to attach to the appropiate node, idea etc in the conceptual or document workspace. (e.g. might have a flash on something I wrote ten minutes ago but don't *now* want to break my train of thought in scroll/paging/flipping back to the correct spot to attach said "flash").

>note how in the above I use bolding to highlight phrases that capture the "esence" of some idea that might well need to be relocated in some other part of the doc space.. I could see an automated system that pulls all such highlighted phrases together into a special browsing window that would enable me to quickly review the material. (Of course, the key phrases would have backwards pointers so that I could touch the phrase/item in the browse window and the text segment that was responsible for generating (or containing) it would leap to the surface in a new, scrollable window.

>Actually I use italics, bolding etc for all kinds of issues, including marking some randomly generated phrase as being a cute phrase or turn-of-phrase that I might want to use in the final paper. We need to think about a taxonomy or epsitomology of such annotations for they would serve as the structuring links of the document space. [Ideal set of issues to see if Master Collins would want to collaborate etc]

ii) Need an **integrated message system** that could enable me to pop off a msg requesting something say from jackie like a reference or an address or a spelling of a name etc. Might want to think about how to have the msg system leave a track behind so that for a certain class of msgs the forthcoming answers could be automatically inserted in the correct place in the document space. (a tech wow wow) Also want reminder cues-- more on that in a moment-- that get established if a msg has been sent out with no reply within a specified amount of time (a field setable on the msg header by the sender. perhaps). Note that we will need to create a taxonomy/epistomology of the differenct kinds of msgs along with their operational significance.

iii) Reminder lists/windows: As Malone's study indicates, our desktop model (maybe) should have slightly visible window/piles of things that haven't been completed yet stemming from various kinds of scratchpads. Or, at least when we near the end of the given session a la cleanup stage we should have the scratch pads elevated to visible status or something like that. [Want windows and icons for each window so that when window is burried, icon reminds me that such windows still exist and are alive.]

>Of course during the cleanup stage or whatever when one is reviewing a Reminder list one wants to be able to spawn new lists and conveniently resort and transfer stuff to it. I suppose every object, list or even text fragment should have some descriptors automatically computed and associated with it that could later be browsed or computed on. For example, a complex time/author stamp should be built that keeps an audit trail of when it was created, modified etc and by whom plus perhaps how large it is etc. (This leads to

some nasty points concerning when is a fragment still a fragment especially when one fragment gets modified by copying pieces of other fragments into it.)

6) Summarizer: (three dimensions: one over the structure of a given layer or version of the doc, another over the small increments or modifications within the current working version--much akin to the notion of a context a la PIE, and the last over the versions themselves. This latter having to do with quickly summarizing what changed, by how much, by whom, to what end {perhaps, what ever that might mean} etc over the course of the major working sessions on the paper as opposed to just the minor editing and updating sessions. Each dimension probably should have its own window and display technique.)

i) Goal: As I am developing a list of points etc such as is happening in the mere writing of this small doc, I often loose track of what numbered point should come next or worse, I want to renumber the issues after skimming backwards to find what the next number should be. What I want is another window/structure that is keeping a "bird's eye view" of points and subpoints etc. The question is how best should a summary structure be displayed and made arbitrarily expandable. Also, how should it be automatically constructed since, perhaps, a lot of the time I don't want to actually systematically number each subpoint. And, if points are numbered then some other set of techniques must be considered to define the scope and embedding level of the point/subpoint.

Some possiblities:

-maintain a **topological structure** where each node contains the opening sentence or an opening compressed structure such as might be obtained by deleting all the function words for the sentence. Might augment this by the last sentence as well plus some indication of the size of entity/object. Of course, as one modified or expanded the point, then the summary node should reflect this fact. *An outline window?*

-might want to have the writer pick a key phrase out of the text that would constitute the content of the summary node or atleast serve as a mneumonic for the content.

-have a frame pop-up for the summary node that the author could indicate what he wants to include but the frame would have a predefined and cognitively motivated set of slots.

-have a **spatial analog** that shows the shape/contour and size of the section/point etc. Of course depending on the size of the document this would have to readjust (by itself) to the right level or grain size.

ii) Updates: For the summarizers to do much "work" we need to have them intimidately cross linked not only with each other but, in particular cross linked to the text so that I can freely move the nodes of the summarizes around and the linked or associated text moves correspondingly and vice versa. Of course this is less of an issue if anything we put into the system is a structured object so that the mere notion of "text" as a stand alone entity makes no sense. (Not really true on second thought because the issue still exists concerning when (and perhaps how) should the objects be interwoven.)

>structured editing vs flat-to-structure a la Emacs/lisp: Seems like there might exist two radically different approaches here. One might start off from Beau's syntax-directed structural editor--i.e. which, in this case, might mean nothing more than having a neutral "form" present itself which the writer then uses as a template within which to embed text (somewhat akin to having a basic template for lisp functions which enables the structural editor to win for entering initial stuff as opposed to just "winning" after substantial structure is already present.) The other approach is is more akin to writing CADR Lisp in Emacs...

7) Assumed Uses: Before pushing much more on all these structural contexts, layers/versions etc we need to lay out what the possible uses or goals of such a system might be along with their corresponding assumptions.

i) Teaching writing: Around such a beast we could design some pretty niffty writing environments for students that could implicitly or explicitly reflect cognitive theories of writing. Here, I'm thinking about Flower and Hays at Cmu who have detailed some of the problem-solving strategies that underly writing and how these "planning" strategies might be embedded into browser like notions to help make kids explicitly aware of them and how to use them. Such an approach is akin to Mark Miller's thesis stuff for teaching kids Logo programming. (If pursued should get a look at Van Nostrum's apparently successful book on Functional Writing along with some classic books on rheotoric--but more on that later.) (Also need to check on what BBN is up to)

ii) studying the writing and/or creative human endeavors: Being able to keep a structured and complete audit trail that captures the *evolution* of a doc or set of ideas could provide fascinating material for a range of cognitive science studies. We have yet to carry forth on Reitman's old dream of having technologly provide a sufficient boost to a scientist that he would be willing to use it to record his daily thoughts etc. This might be our first realistic chance to not only create such a beast but to have it

sufficiently cost effective that it could be widely used!

[a reasonable notion, considering the ancient and honorable history of notebooks.]

iii) writing and reading: there are several dimensions here--

-single writer vs multiple authors

-single purpose document or multiple purpose in the sense that I might want to build up one document space from which I could generate a short paper, a long paper, a set of notes, slides for a talk to different kinds of audiences etc.

-single or multiple readers who are going over a preliminary version of the paper. What kind of annotation facilities should be provided that might help pull together multiple sets of annotations, each coming from a different preliminary reader.

[>refer here to issues of having the base editor enable one to make comments on a paper much as he would do with a pencil and paper--namely cram annotations into all the available whitespace while keeping the spatial layout of the surface/print doc invariant. There is much to say here but I think I should just dig up the old working paper I wrote just on this issue--but actually this issue is more important if all our reading material is electronic.]

-presume that all the joint writers and/or readers have or do not have access to an authoringland system. Assuming that that all the writers/readers do have access, then what is getting passed for comments might be a) still just the surface doc or b) the top layer of the workspace with all of its summary structures etc for enabling structured browsing or c) a set of accessible layers...

8) Annotations: We need to have an epistomolgy or taxonomy of the *kinds* of things we want to say about the *kinds* of things we are working on.

Concerning the latter in the midst of writing/brainstorming/flaming etc we have "ideas" or beliefs that concern:

-the essence or substance of the doc

-the form or structure of the doc(e.g. size, outline...)

-the form or structure of an subobject that is to appear in the doc such as a figure, program, graph, math equation etc.

-actual phrases, figures, cartoons, jokes, puns etc that might occur in the surface structure or printed form of the doc

- the intended audience or readership of the doc (management-high vs middle, scientific, computerists etc.)

- style and plans for communicating the ideas (e.g. folksy vs technical, placid vs agressive, etc)

-etc

Concerning the former (and its not at all clear how different, in principle, the former are from the latter) we might have commentary about the idea or belief in the form of:

-what is it intended to achieve or communicate. (mainly for such annotating such things as figures, diagrams etc where tnis could be taken, almost, as a caption)

-do I believe it

-what are its weaknesses

-what are its strengths

-how should it be expanded or what is left to be done on it

-what might be some good/poor examples of it

-how might it be misunderstood

-what are its pre and post requistes (e.g. don't try to explain it until you have covered xxx)

-could be left off for a short version

-it's a bad idea but might reflect how someone might interpret what is being said (a kind of thing that often emerges as one is working thru some subtlety and recognizes that if the reader "processes" the argument only so far he might think thusly e.g. a potential *mindbug.*

-arguments, counter-arguements and counter-counter args and so on to an "idea" which of course recurses.

There is much work to be done just on this simple epistomology and the way to get started is probably just to build a system that can enable us to specify *tags* for each kind of entry and see what we can get the system to do with such tags--do the tags/distinctions help us browse, reorganize, remember, restructure, collapse, create multiple versions for multiple purposes etc. That is, do the tags have well-defined operational meaning that enables us to write operators for AuthoringLand that help in the above tasks.

(The above paragraphs exclude here all issues concerning what the kinds of comments or annotations are that we might want to include for information to be passed to the doc compiler/formatter etc.)

Perhaps we should just start, initially, by creating a authorland that helps us, the designers, to write, to think, to flame, etc. and then later worry about its extendability to other writers and then to other tasks such as education/literacy aids . . ., decision aids, etc. I believe that the current Interlisp-D machines/environments now contain most of the necessary tools to create a prototype system quite easily.